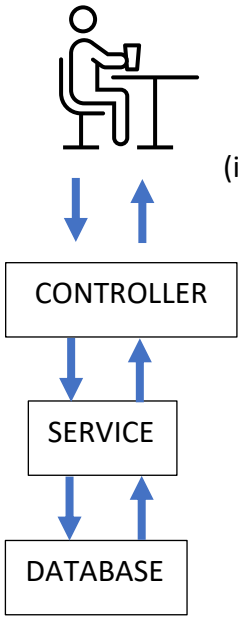
Below are the steps you need to take to have the application running on your machine. Keep in mind the layered structured of the application, and the flow of the data within:



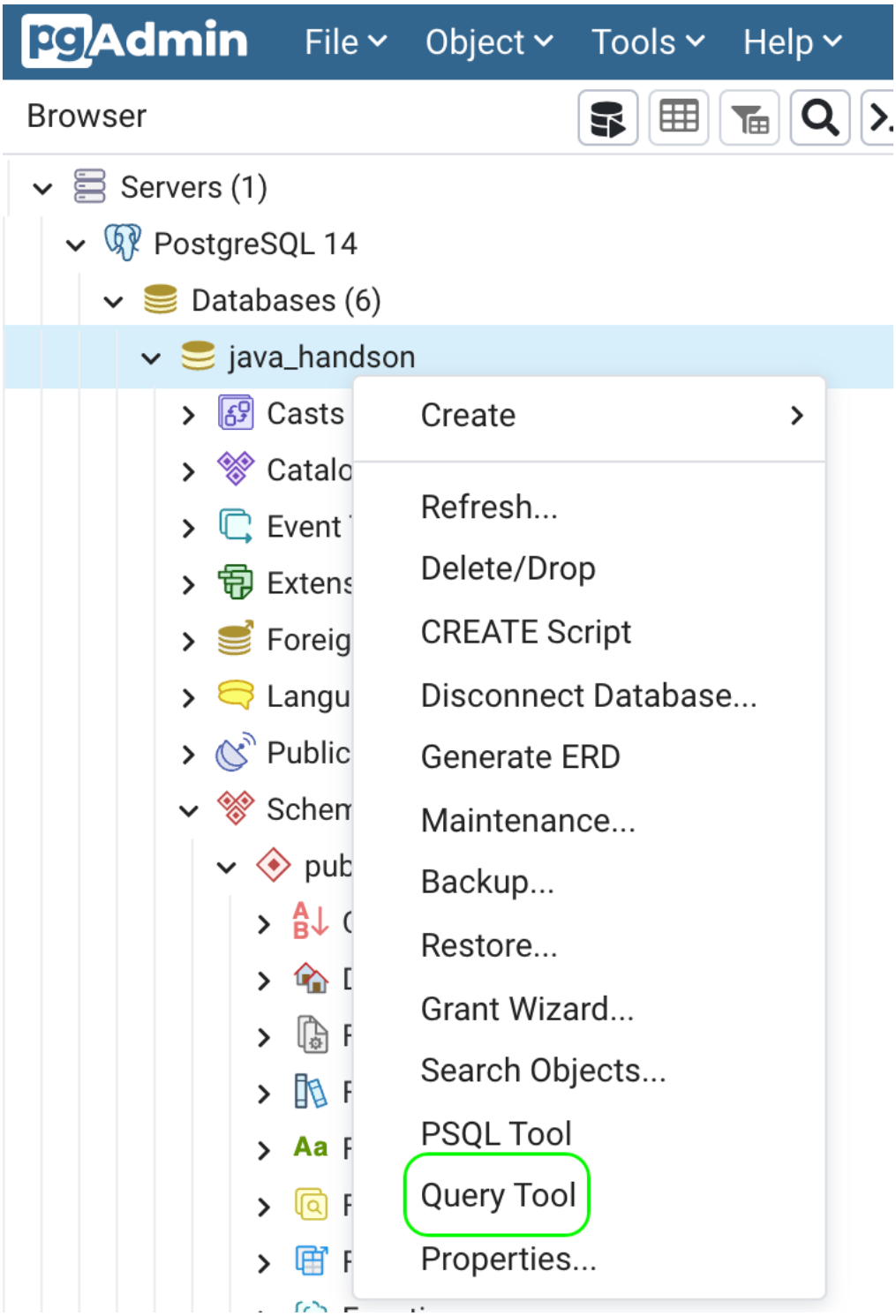
(in our case – Postman)

Unzip the file and follow the instructions. Feel free to ask questions if you want.

I. Steps for preparing the DB (for those who already did it in the classroom, you can skip):

1) In any query tool run: `CREATE DATABASE java_handson;`

2) Open a query tool from pgAdmin for the new java_handson DB:



3) In the new query tool add and run:

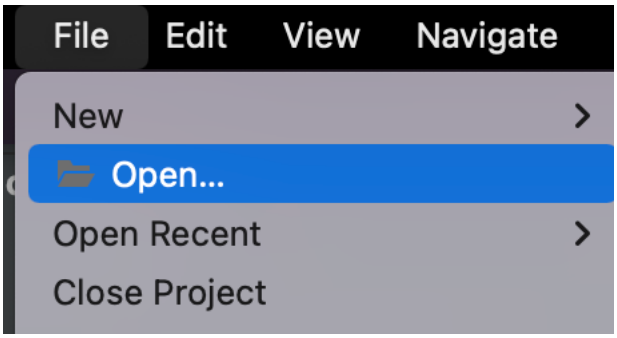
```
CREATE TABLE employees
 (
     emp_id SERIAL PRIMARY KEY,
   birth_date date NOT NULL,
     first_name character varying(255),
     last_name character varying(255),
     hire_date date NOT NULL
 );
```

II. Steps for loading (in order to look over the code) and running the employees-component in IntelliJ CE:

1) Download the zip

2) Unzip and copy the 'employee-component' directory to a place of choice

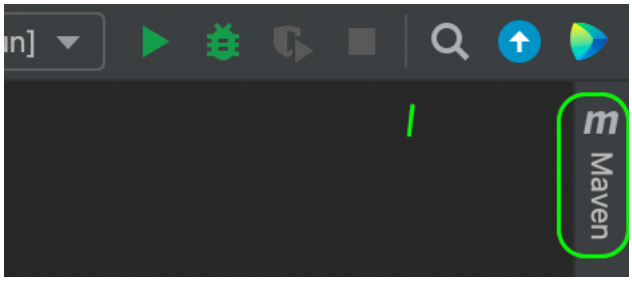
3) Open the project from IntelliJ CE



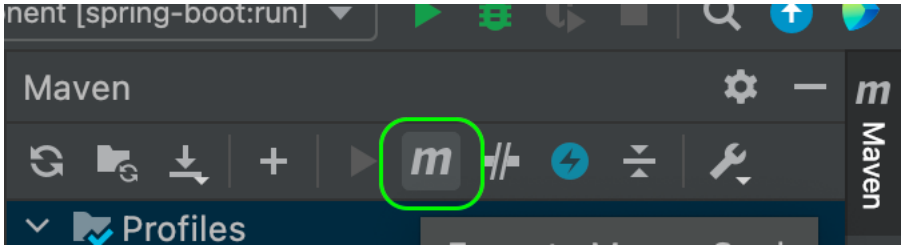
Navigate to the above 'place of choice', and from 'employee-component' directory select **pom.xml** and choose to open it **as a project** (not as a file)

4) Build the project:

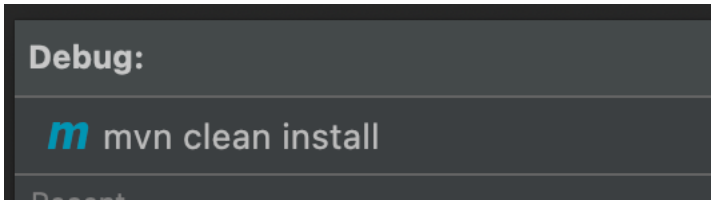
   a. On the right side click on **Maven**



b. Click on the **m**

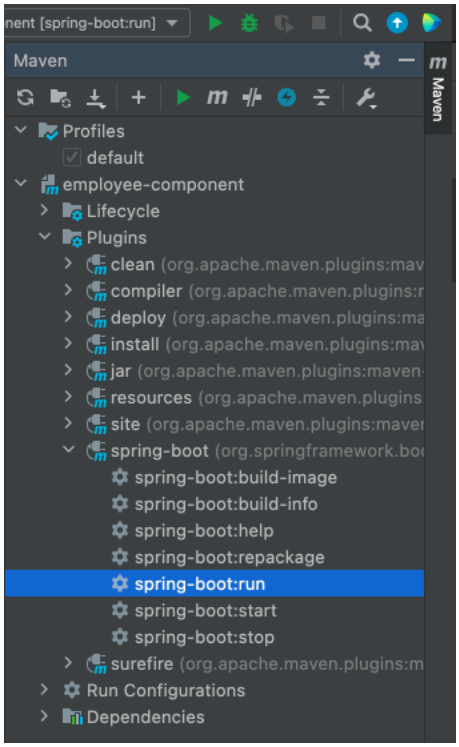


c. Type: `mvn clean install` (then press enter)

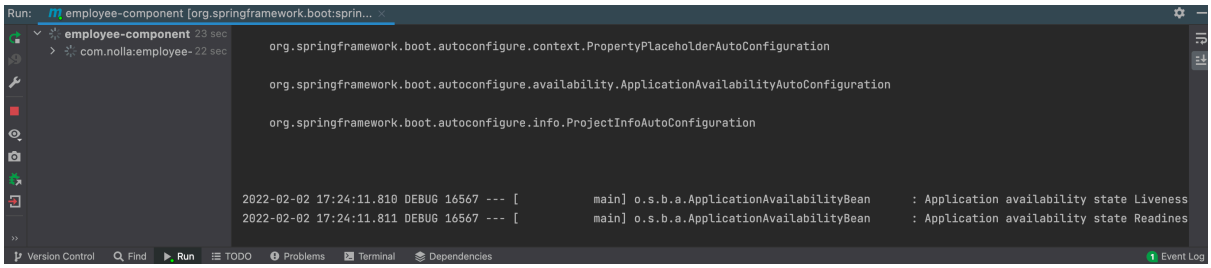


5) Run the project:

a. From the same **maven**, go to **employee-component -> Plugins** and double click on **spring-boot:run** (this is the maven command which will start the application)



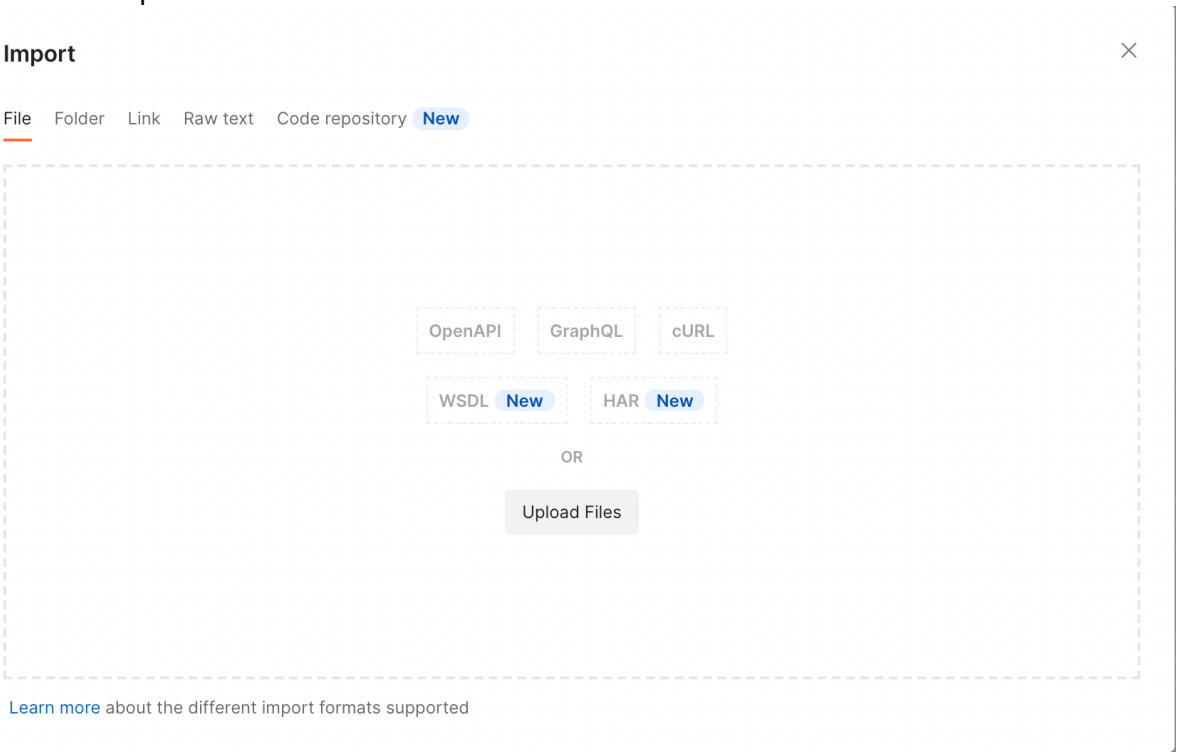
b. Successful run should look like:



III. Steps for testing with Postman:

III.1 Import testing collection:

1) Download the postman collection (json file)
2) File -> Import

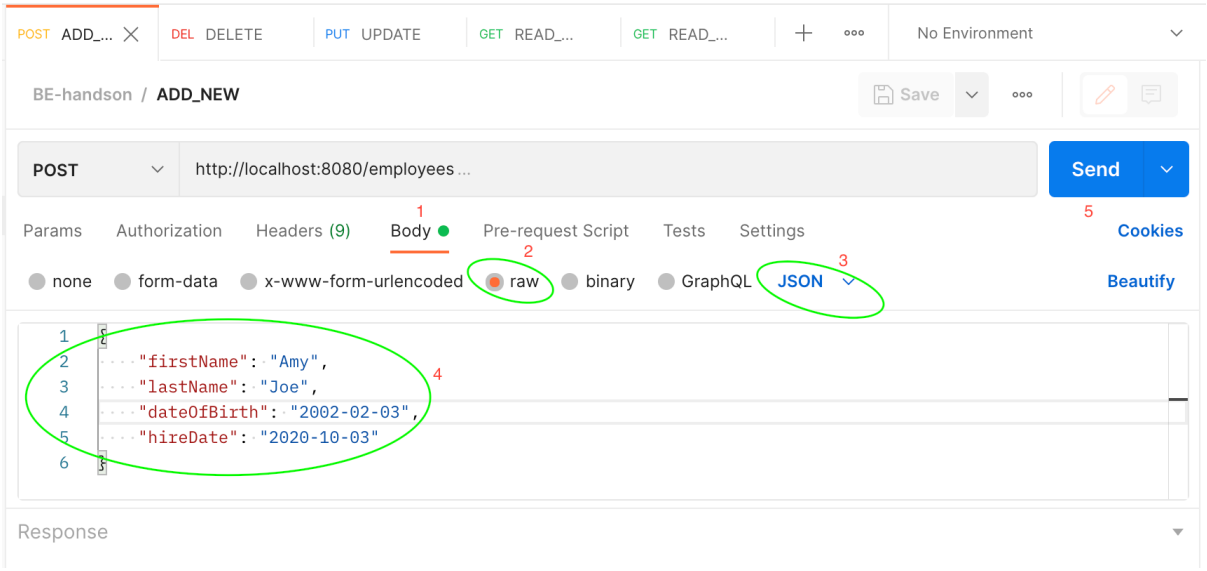


3) Import it via [Upload Files]

III.2 Use the imported calls:

! These calls will be handled by your controller class.

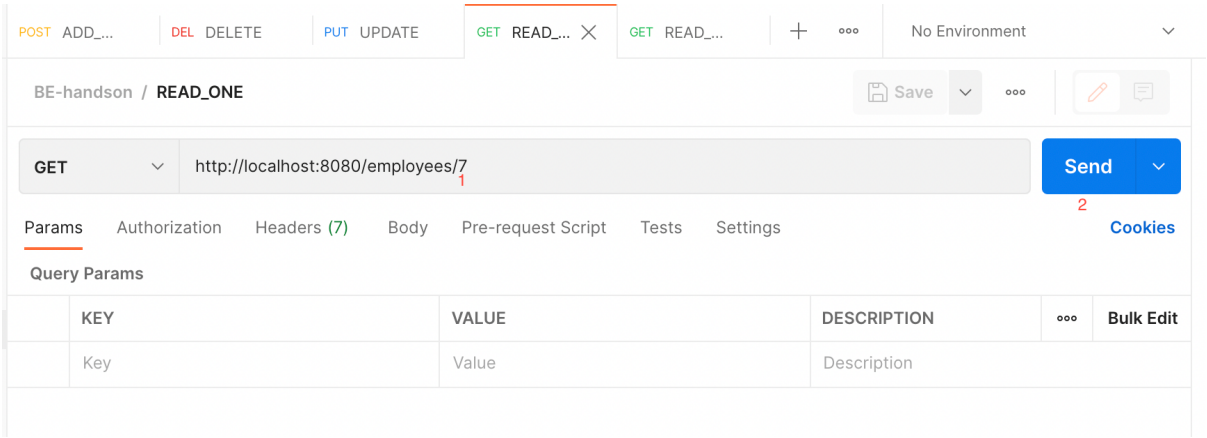
CREATE (use it first to you populate the table which you can check in the **pgAdmin** using a SELECT statement)



Request Body sample:

```
{
    "firstName": "Amy",
    "lastName": "Joe",
    "dateOfBirth": "2002-02-03",
    "hireDate": "2020-10-03"
}
```

READ (there are two end-points (by end-point you can think of one functionality exposed by the API, as we've discussed in class): READ by emp_id and READ ALL



UPDATE you can relate to CREATE approach as it is similar

DELETE